Hardware implementation of an event-based message passing graphical model network

Chien, Chen-Han ; Longinotti, Luca ; Steimer, Andreas ; Liu, Shih-Chii

Abstract: This paper presents a hardware system that implements a factor graph, where messages are sent using an event-based belief propagation algorithm. The system, comprising an FPGA and an application specific integrated circuit (ASIC) chip, can be used to construct a graph with upto 16 output message channels. The ASIC chip with 16 channels is fabricated in a 0.35 um 2P4M CMOS process and occupies 2.16 × 2.74 mm 2 . Each channel dissipates 46 uW. The output analog messages of the channels are encoded through the interspike intervals of the output spike streams or events. The system can be used to implement graphs with arbitrary variable distributions for its inputs and using constraint functions, such as “plus” and “equality”. Using Kullback-Leibler divergence, we show that the measured distributions from the implemented graphs on the hardware show close similarity to the theoretical distributions.

DOI: https://doi.org/10.1109/TCSI.2018.2798289

Hardware Implementation of an Event-Based Message Passing Graphical Model Network

Chen-Han Chien, Luca Longinotti, Andreas Steimer, and Shih-Chii Liu, Senior *Member, IEEE*

***Abstract*—This work presents a hardware system that implements a factor graph where messages are sent using an event-based belief propagation algorithm. The system, comprising an FPGA and an ASIC chip, can be used to construct a graph with up to 16 output message channels. The ASIC chip with 16 channels is fabricated in a 0.35 um 2P4M CMOS process and occupies 2.16 × 2.74 mm^2. Each channel dissipates 46 uW. The output analog messages of the channels are encoded through the interspike intervals of the output spike streams or events. The system can be used to implement graphs with arbitrary variable distributions for its inputs and using constraint functions such as "plus" and "equality". Using Kullback-Leibler divergence, we show that the measured distributions from the implemented graphs on the hardware show close similarity to the theoretical distributions.**

***Index Terms*— factor graph, hazard function, random sampling, renewal theory, interspike interval, event-based, spike-based, real-time**

I. Introduction

The increasing availability of different spiking neural network hardware platforms that are implemented through either custom mixed-mode analog/digital or digital VLSI spiking neuron arrays or on FPGA [1]–[6] have allowed the validation of various neuroscience and machine learning spiking models for practical systems. The use of stochastic models on these neuromorphic spiking platforms is still relatively scarce although stochasticity by itself could be useful, e.g. to decorrelate the firing of neurons in a population [7]. There is however an increasing interest in sample-based stochastic networks such as Deep Belief Networks using layers of Restricted Boltzmann Machines (RBMs). Spiking versions of RBMs [8] have been tested on applications such as image recognition using the MNIST dataset and in a sensory fusion task using the Dynamic Vision Sensor [9] and Dynamic Audio silicon cochlea Sensor [10]. The inference in these networks is done using a Markov Chain Monte Carlo procedure called Gibbs sampling. This stochastic spiking network can also be trained on-line by using an event-driven approach of the training method for the RBM [11]. A spiking RBM has been mapped on the neuromorphic TrueNorth system with digital spiking neurons and by using a noisy threshold model to implement the Gibbs sampler [12] and a spiking DBN was implemented on the FPGA using inputs encoded as unary streams [13].

Another spiking inference model is the event-based factor graph model proposed in [14]. Factor graphs are one instantiation of graphical models where the nodes of the graph represent functions of subsets of variables; and messages are transferred between the nodes using methods such as belief-propagation [15], [16]. This event-based model uses a temporal coding scheme based on interspike intervals (ISIs) in spike trains to communicate messages between the nodes of the graph. This scheme is inspired by biological studies that suggest neurons could communicate information by using ISIs [17]–[19].

In [14], a spike is treated as a random sample, whose numerical value is given by the spikes preceding ISI. In this model, spike trains are assumed to follow renewal processes and hence to correspond to sequences of independent random numbers, that is, sequences of labeled spike events, where each spike's label corresponds to an ISI random number as shown in Fig. 3(a).

Because this stochastic model cannot be easily implemented on the currently available spiking network hardware platforms, an explicit implementation of the event-based message passing scheme was presented in [20]. This VLSI implementation is based on direct correspondence between fundamental equations from renewal theory and the physical behavior of the analog Very Large Scale Integrated (aVLSI) circuit elements. The principle allows for the generation of sequences of arbitrarily distributed random numbers that are confined to the positive real axis. In the work presented here, the VLSI message-passing circuit in [20] is extended to an Application Specific Integrated Circuit (ASIC) chip with an array of 16 channels that produce output messages. The calculations of the analog messages carried by the ISIs of the input spike trains and the output of the factor functions are carried out using an FPGA for flexibility in constructing graphs with different factor functions. This work describes the circuit details of the ASIC chip in a 2-poly-4-metal 0.35um CMOS process. It also presents measurements from the combined ASIC + FPGA system used in the

This work was supported in part by the Swiss National Foundation Stochastic Event Inference Processor Grant, SNF grant #200021_135066. C.-H. Chien, and S.-C. Liu are with the Institute of Neuroinformatics, University of Zurich and ETH Zurich, Winterthurerstrasse 190, Zurich, Switzerland (e-mail: chenhan@ini.ethz.ch, shih@ini.ethz.ch.).

L. Longinotti is with iniLabs Ltd., Zurich, Switzerland (e-mail: luca.longinotti@inilabs.com).

A. Steimer. is with the Bosch Center for Artificial Intelligence, Renningen, Germany (e-mail: Andreas.Steimer@de.bosch.com).

construction of example factor graphs.

The paper is organized as follows: Section II describes the theory of the event-based belief propagation model. Section III describes the details of the implemented hardware system. Section IV describes results from this hardware system. Section V presents discussions on this work.

II. Event-Based Graphical Model

A. Forney Factor Graph

We first introduce some basic notions of the Forney Factor Graph (FFG) that are useful for the description of our work. More details of FFGs can be found in [15], [16]. A FFG is a graph-based representation of a factorized joint probability distribution, such that the nodes of the graph correspond to the nonnegative factors of the factorization. The edges in turn correspond to those variables, on which the factor functions they are connected to, depend on. An example of a factor graph representing the factorized joint probability of the variables described in (1), is shown in Fig. 1. The nodes of the graph correspond to the individual factors (f_1 to f_6) of the factorization in (1) and the edges correspond to the variables.

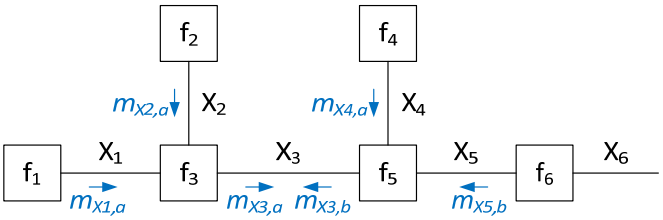


Fig. 1. A factor graph generated from the factorized joint probability in (1). The functions are defined as nodes and the variables are represented as edges. The description of the messages (blue fonts and arrows) are described in the main text.

$$f(x_1,..,x_6) = f_1(x_1) f_2(x_2) f_3(x_1,x_2,x_3) \\ f_4(x_4) f_5(x_3,x_4,x_5) f_6(x_5,x_6) \quad (1)$$

We assume for the remainder of this paper that the variables are discrete and therefore only summations, rather than integrations, are needed for marginalization. The marginal probability of each variable can be computed by summing over all variables except for the desired variable. The marginal probability, e.g. of X_3, is shown in (2) with a normalizing factor described in (3).

$$p(x_3) = \sum_{\substack{x_1,\ldots,x_6 \\ \text{except } x_3}} f(x_1,\ldots,x_6) / Norm \quad (2)$$

$$Norm = \sum_{x_1,\ldots,x_6} f(x_1,\ldots,x_6) \quad (3)$$

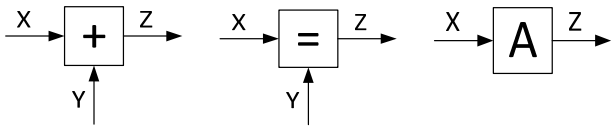
Using the belief-propagation approach, marginalization can be performed a more efficient way than in (2), by exchanging 'messages' between adjacent nodes along the connecting edge (variable). These messages can be interpreted as probability mass functions that depend on the connecting variable and, for the graph in Fig. 1, are computed following the set of equations in (4). As Fig. 1 shows, a message m (blue arrows) sent from one factor to one of its neighbors is formed by the product of all input messages into the sending node (except for the message coming in along the same edge as m) and the factor function represented by the sending node. The resulting product is then summed across all variables connected to the sending node, except the variable (edge) the output message is passed along. The method for computing output messages in this way is called the sum-product rule (SPR). Note that in (4) the output message of each equation is equal to its right side up to a scale factor $Norm_{Xi}$, which is formed by summing along the output variable X_i.

$$\begin{aligned} m_{X1,a}(x_1) &\propto f_1(x_1) \\ m_{X2,a}(x_2) &\propto f_2(x_2) \\ m_{X3,a}(x_3) &\propto \sum_{x_1,x_2} f_3(x_1,x_2,x_3) m_{X1,a}(x_1) m_{X2,a}(x_2) \\ m_{X3,b}(x_3) &\propto \sum_{x_4,x_5} f_5(x_3,x_4,x_5) m_{X4,a}(x_4) m_{X5,b}(x_5) \\ m_{X4,a}(x_4) &\propto f_4(x_4) \\ m_{X5,b}(x_5) &\propto \sum_{x_6} f_6(x_5,x_6) \end{aligned} \quad (4)$$

To obtain the marginal probability of X_3 in Fig.1, we only need to multiply the messages from the left and right sides of the edge associated with X_3 as follows:

$$p(x_3) \propto m_{X3,a}(x_3) m_{X3,b}(x_3) \quad (5)$$

Although the belief propagation approach presents an advantage in that the bidirectional messages on all edges are formed by summations across only a subset of all variables and can be computed in parallel, the massive cost involved in computing the SPR is still required. This cost can be reduced by considering only Gaussian distributed variables and restricting the defined constraint functions to a few types, e.g. **plus**, **equality**, and **gain** as shown in [15]. Factor nodes for these functions (see Fig. 2) are described in (6), and the unidirectional output messages, for instance, are computed as shown in (7). As a result, the mean and variance of the distributions are the only parameters needed during message passing and the message computation rules can easily be tabulated. This constraint reduces the computational load. However, for messages of arbitrary distributions and for more complex user-defined functions, the full calculation of the SPR is needed.



(a) (b) (c)

Fig. 2. Symbols of specific factor nodes. The arrow indicates the direction of message passing. The nodes define the (a) plus, (b) equality, and (c) gain constraint functions with variables X, Y and Z.

$$\begin{aligned} f_{plus}(x,y,z) &= \delta(z-(x+y)) \\ f_{equality}(x,y,z) &= \delta(x-y)\delta(y-z) \\ f_{gain}(x,z) &= \delta(Ax-z) \end{aligned} \quad (6)$$

$$
\begin{aligned}
m_Z(z) &\propto \sum_{x,y} f_{plus}(x,y,z)\, m_X(x)\, m_Y(y) \\
m_Z(z) &\propto \sum_{x,y} f_{equality}(x,y,z)\, m_X(x)\, m_Y(y) \\
&= m_X(z)\, m_Y(z) \\
m_Z(z) &\propto \sum_{x} f_{gain}(x,z)\, m_X(x) = m_X\left(\frac{z}{A}\right)
\end{aligned}
\tag{7}
$$

B. Event-Based Belief-Propagation Model

Instead of representing the SPR as a list of probabilities, the work in [14] showed how analog messages can be represented as a list of ISIs of any two consecutive events in the input and output spike streams of the factor nodes. This model avoids the normal expensive SPR computation. That is, the SPR summations are solved in an implicit way by means of Monte Carlo sampling. In addition, the factor graph in this formulation is not limited to the use of Gaussian messages. The event-based belief propagation approach bears some similarity to the sequential Monte Carlo sampling method [21], [22], where many particles are used to approximate the sampled input distribution. This message passing formulation is inspired by experimental evidence that shows that populations of neurons can be sensitive to the timing of their inputs and that the input to a neuron can depend on the spike input frequency or the difference between the arrival time of spikes [23], [24]. The mechanism of the event-based belief-propagation model in [14] is briefly explained here:

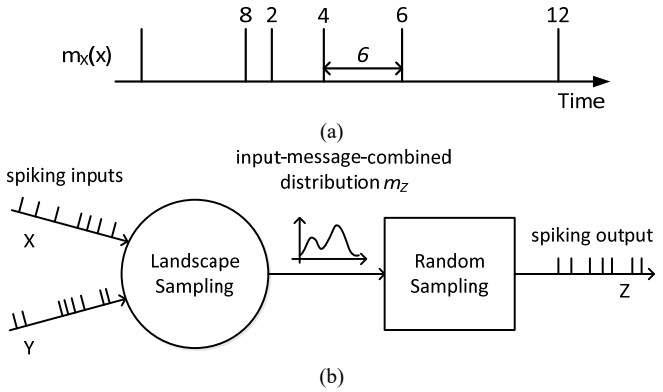


Fig. 3. Proposed message encoding and passing scheme. (a) Each spike carries an analog label with a value corresponding to the ISI preceding the spike. (b) The factor's function *f* is implemented in a landscape sampling block which produces the probability distribution of message, m_Z, used by the random sampling block to generate the spiking output.

1. The message is encoded in the ISIs of the spikes or events as shown in Fig. 3(a). Given a probability distribution representing a message, each ISI value is a random sample from this distribution. Within a finite time window, *W*, the statistics (empirical distribution) of the samples approximates the true ISI distribution representing the message.

2. A unidirectional message passing is composed of a landscape sampling (LS) block and a random sampling (RS) block as shown in Fig. 3(b). We show only two input variables in the figure but there is no limit on the number of inputs. A complete factor node for this example would be made up of three such circuits in order to compute the messages in both directions for all three variables. In the LS block, samples consist of pairs of the most recent input ISIs of the two variables *X* and *Y*. These samples *(x,y)* are both sent to the factor's function, *f(x,y,z)* and the summation function *F(x,y)*. The function *F(x,y)* is a sum of *f(x,y,z)* with respect to the output variable, *Z*, and is used as a normalizing term. Once all pairs in *W* are computed, the input-message-combined distribution m_Z is computed from the histogram of *f(x,y,z)* normalized by the value of *F(x,y)*. The message m_Z is then sent to the RS block that generates the output spikes.

3. For the ISI distribution in the output spike train to approximate m_Z, the computation in RS is as follows: First, we use the relationship between the given probability distribution *p(t),* and the hazard value *h(t)* which can be interpreted as a conditional instantaneous firing rate, assuming that the last spike occurred at $t = 0$. The expression of the hazard function is described in (8) and given by [14], [25].

$$
h(t) = p(t) \cdot \exp\left(\int_0^t h(t')\,dt'\right) \tag{8}
$$

Second, by using the discrete time approximation, random numbers *nx(t)* are sampled uniformly in the range of [0,1] at a discrete time step, *Δt*. Here *Δt* is defined as 1 so *t* increases in integer steps. Given that no spike has occurred in time 1 to (*k*-1), the probability of generating a spike at time *k*, is equal to the probability that $h(t) > \frac{nx(t)}{\Delta t}$ when $t = k$ and keep $h(t) <= \frac{nx(t)}{\Delta t}$ during $t = 1{:}(k\text{-}1)$. Otherwise, *h* continues accumulating over time until an event is generated.

If *X* and *Y* inputs to the LS block are independent of each other and *W* is sufficiently long, the paired ISI samples would approximate the input messages m_X, m_Y. Also, the distribution of the output ISI samples of *Z* would approximate m_Z.

III. Hardware Implementation

A. System Architecture

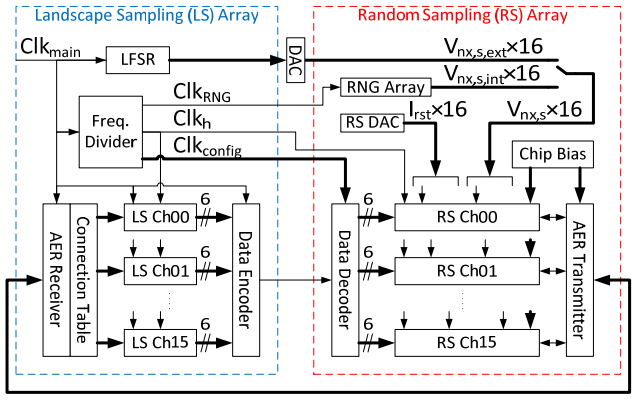


Fig. 4. System architecture consists of two blocks. Left (dotted blue box), the landscape sampling block with a 16-channel landscape sampling (LS) array and right (dotted red box), the random sampling block also with a 16-channel random sampling (RS) array.

The hardware implementation of the system (Fig. 4) follows the system architecture in Fig. 3. The LS block is implemented on an FPGA for flexibility and the RS block is implemented as

an ASIC designed in a 0.35um 2-poly 4-metal CMOS process. With the 16 RS channels on the ASIC chip and the LS channels on the FPGA, factor graphs with up to a maximum of 16 output messages can be configured using this system. The communication from the RS block to the LS block is via the **AER Transmitter** block in the RS and the **AER Receiver** block in the LS and using the asynchronous address event representation (AER) protocol [26]. The connections between the **AER Receiver** and the LS channels are defined in the **Connection Table** block of the FPGA. The **Data Encoder** block combines the messages (m_Z) of the individual LS channels into a single output of pulses transmitted consecutively from each channel (see Section III.B for details). This single train of output pulses is then transmitted to the **Data Decoder** block within the RS chip. This block remaps the output spike trains to the corresponding RS channels. The system also has two possible sources of random number generators. The first is through the linear feedback shift registers (LFSRs) and off-chip digital-to-analog converters (DACs) which convert the digital output of the LFSR to analog signals, $V_{nx,s,ext}$, and the second is a pseudo random number generator (RNG) [27] array on the chip providing individual uniform random variables, $V_{nx,s,int}$, for the 16 RS channels. The LS block uses the main FPGA clock (Clk_{main}) for the FPGA modules and also generates three different clocks (Clk_h for RS channels, Clk_{RNG} for RNG array, and Clk_{config} for Data Decoder) using a frequency-divider module. The LS block uses the FPGA clock Clk_{main} to accelerate counting of the ISI statistics and the RS chip uses a frequency divided-down clock Clk_h needed for the longer time constants of the aVLSI circuits.

B. Structure of Landscape Sampling Channel

Fig. 5 shows the structure of one LS channel. The building blocks and signal flow between the blocks follow the basic mechanism described in Section II.B. The LS channel is designed for a two-input-one-output or a one-input-one-output message passing. We limit the input number to a maximum of two in our implementation to reduce the complexity of the structure. However, the input number can be increased in future implementations.

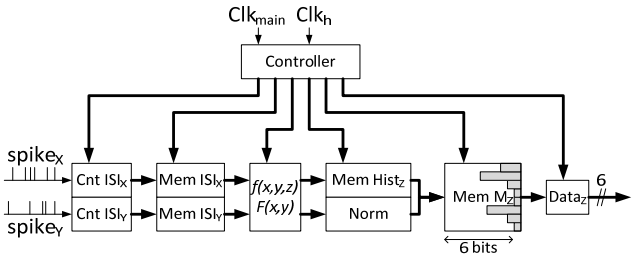


Fig. 5. Structure of one Landscape Sampling channel

The asynchronous spike trains from the **AER Receiver** block carry the messages $m_X(x)$ and $m_Y(y)$, of the two variables, X and Y. The **Cnt ISIX** and **Cnt ISIY** modules in Fig. 5 measure the ISIs of the spikes in the event streams of X and Y respectively. The module starts a counter as soon as a spike arrives. The counter increments in unit time steps, Δt, and the counter stops when the next spike arrives. The measured ISI value is then stored in the memory modules, **Mem ISIX** and **Mem ISIY**, respectively. After W defined in the **Controller** module has expired, the ISIs stored in the memory modules are transferred to the **Function** module which is programmed with the desired factor's function $f(x,y,z)$ and summation function $F(x,y)$. The function $f(x,y,z)$ in this work is limited to delta functions. To ensure enough input samples for extracting the statistics of the distribution, sample pairs (x,y) are generated from all combinations of ISIs in the **Mem ISIX** and **Mem ISIY** modules. This method is slightly different from the pairing method described in [14], where the samples consist of only pairings of the latest ISI values. The histogram counts of $f(x,y,z)$ and of $F(x,y)$ for normalization are saved to the **Mem HistZ** and **Norm** modules, respectively. We have m_Z stored in the **Mem MZ** module after normalization. The index number of ISI bins, ind, is in the range of ISI values varies from (ind = 1) to ($ind = ISI_{max}$). The value in each m_Z bin is represented by 6 bits, which is the width of the **Mem MZ** module. The picture in the **Mem MZ** module as shown in Fig. 5 demonstrates an example of a possible probability distribution of the message m_Z. To consume less FPGA resource, the memory values in **Mem MZ** module only updates when the count of $F(x,y)$ reaches 2^k each time, where k is an increasing integer. In this way, normalization only involves a bit shift. In the initial state of the system and at the beginning of each time window, the **Mem ISI**, **Mem HistZ**, and **Mem MZ** modules are reset. The **DataZ** module transmits the value of the individual m_Z bin in a sequential manner starting from (ind = 1) to ($ind = ISI_{max}$) to an RS channel. Once an output spike is generated from RS and sent back to LS by AER request, the **DataZ** module re-starts to transmit the bin value from (ind = 1) regardless of the position of ind before the spike.

Note that the data transmitted by **DataZ** module change discretely every Δt. Therefore, ISI_{max} is always an integer value while t_{ISImax} defined as ($ISI_{max} \times \Delta t$) represents the actual ISI time in seconds. This definition is adopted in the paper.

C. Structure of Random Sampling Channel

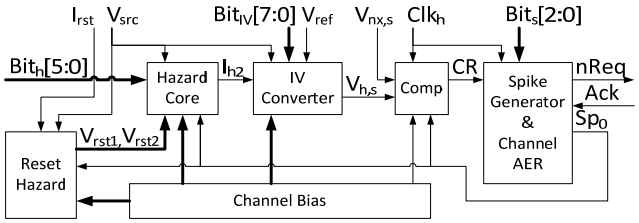


Fig. 6. Structure of one Random Sampling channel

The RS chip generates output spikes according to the message m_Z. It comes from the **DataZ** module of the LS, transmitting the value of the individual m_Z bin sequentially. The 6-bit data in each bin is represented as $Bit_h[5:0]$ as shown in Fig. 6. As described in Section II.B, the mechanism requires a hazard function, a comparator, and a reset. The **Hazard Core**, **Comp** and **Reset Hazard** blocks in Fig. 6 implement these functions respectively. Because the output of the **Hazard Core** is a current in the aVLSI implementation while the comparator in the **Comp** block is a voltage-input comparator, an **IV Converter** block is needed. In addition, a channel AER

represented as the **Spike Generator & Channel AER** block in each RS channel is also required. This block not only communicates with the **Chip AER Transmitter** block, it also controls the pulse width of the feedback spike signal, Sp_0. The details of some blocks are explained as follows.

1) Hazard Core

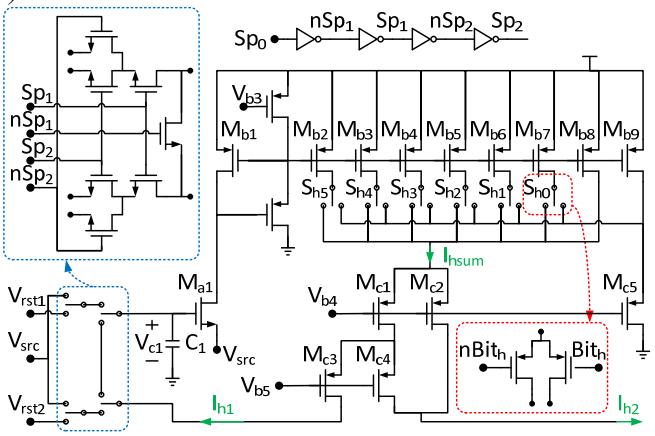


Fig. 7. CMOS circuit details of the Hazard Core block

To realize the **Hazard Core** block, the implementation of [20] uses a capacitor for the integration, a subthreshold transistor for the exponential term, and a translinear loop (TLL) current-mode multiplier circuit [28] for the multiplication in (9) . The VLSI circuit variables in the mapping equation presented in [20] are associated with the hazard function as follows:

$$I_h(t) = I_1 \cdot \frac{I_p(t)}{I_L} \cdot \exp\left(\frac{\kappa}{CU_T}\int_0^t I_h(t')dt'\right) \tag{9}$$

where I_h and I_p are the currents which are proportional to h and p respectively, I_L is a constant bias current, I_1 is the off current of a transistor, κ is the gate-coupling coefficient, C is a capacitor and U_T is the thermal voltage.

The circuit details of the **Hazard Core** block are shown in Fig. 7. Similar to [20], the capacitor C_1 is used for the integration term and the transistor M_{a1} operates in the subthreshold regime to fulfill the exponential term. In this design, we improved on the multiplier circuit in [20] which had a limited input linear range and a varying output current range due to process mismatch. Therefore, the input current range of this previous circuit is hard to define for a set of RS channels on a RS array.

The improvements are carried out in two ways: First we note that $I_p(t)/I_L$ in (9) can be treated as a time-varying unit-less factor $N(t)$. Second, because the LS block is implemented in FPGA and a random sample is only provided on a unit time step, Δt, we can present a new digital input to the RS channel $N(t)$ that is then converted to an analog current through a current-mode 6-bit DAC. By including a DAC in the **Hazard Core** block, we could increase the reliability of the new circuit implementation.

The current-mode 6-bit DAC consists of switches S_{h5} to S_{h0} and transistors M_{b1} to M_{b9} with transistor size ratios of M_{b1} to M_{b9} are 2:32:16:8:4:2:1:1:1. The switches are controlled by a 6-bit RS input, portrayed as $Bit_h[5:0]$ in Fig. 6. The currents in the branches with high input bits of Bit_h sum into a current splitter composed of transistors M_{c1} to M_{c4}. The remaining currents flow through transistor M_{c5}. When all input bits are low ($Bit_h = 0$), the summed current, I_{hsum}, to the splitter shrinks down to off-current level, resulting in a large time constant and slowing the speed. A transistor, M_{b8}, providing a default current to the summed current is added to prevent this situation. The transistor, M_{b9}, is also added to deal with the case when all input bits are high. Thus, including the default current from M_{b8}, the possible 6-bit values are shifted from [0,63] to [1,64]. That is, the time-varying factor $N = Bit_h+1$.

The current splitter ratio of the currents through M_{c1} and M_{c2} and through M_{c3} and M_{c4} is 1:7. The summed current, I_{hsum}, is divided into I_{h1}, the feedback for integration, and I_{h2} for the **IV Converter** block. Due to the current splitter ratio, I_{h2} is $63\times I_{h1}$. This formulation simplifies the design of the **IV Converter** block which will be explained in next section.

The other switch group in the blue box of Fig. 7 controls the integrated voltage on C_1, V_{c1}, and therefore the drain current of M_{a1}. When there is no spike, I_{h1} charges C_1 continuously. Once a spike, Sp_0, happens through the **Spike Generator & Channel AER** block, the feedback loop is opened through Sp_1, Sp_2, nSp_1 and nSp_2. The latter signals are derived from the Sp_0 using inverters with different timing delays in order to minimize the charge injection effect on C_1.

The voltage on C_1 is reset to V_{rst1} and I_{h1} is shorted to V_{rst2}, provided from the **Reset Hazard** block. Both V_{rst1} and V_{rst2} are generated by an input reset current I_{rst} as depicted in Fig. 6. Therefore I_{rst} sets the reset value of V_{c1}, and therefore the initial drain current of M_{a1}. A small initial current corresponds to a longer integration time to reach the same current value. Also, a small input N results in a smaller I_{hsum} and a longer time to reach the same current value. Therefore, t_{ISImax} ($= ISI_{max}\times\Delta t$) is defined both by I_{rst} and N.

In our design, I_{rst} is set to ~300 pA so that I_{h1} at ($N = 1$) as an extreme case, is only a few pA. This setting not only reduces the final power consumption of the circuit but keeps V_{c1} lower than the threshold voltage of M_{a1} before reset. This condition guarantees that the subthreshold voltage-to-current exponential equation is valid. In addition, an offset voltage, V_{src}, is added [29] to decrease the effect of the off currents from the switches during integration and to increase the accuracy of the initial current of M_{a1} during reset.

The hardware mapping equation of the circuit in Fig. 7 is as follows:

$$I_{h1}(t) = \left(\frac{I_{rst}}{128}\right) \cdot N(t) \cdot \exp\left(\frac{\kappa_{M_{a1}}}{C_1 U_T}\int_0^t I_{h1}(t')dt'\right) \tag{10}$$

where $\kappa_{M_{a1}}$ is the gate-coupling coefficient of M_{a1}, $N(t)$ denotes the dynamic ratio input [1,64] and the denominator of 128 caused by the transistor sizing ratio of M_{b1} to M_{b9} and the current splitter. I_{h1} and N are proportional to h and p respectively through

$$h(t) = \frac{\kappa_{M_{a1}}}{C_1 U_T} \cdot I_{h1}(t) \tag{11}$$

$$p(t) = \frac{\kappa_{M_{a1}}}{C_1 U_T} \cdot \frac{I_{rst}}{128} \cdot N(t) \tag{12}$$

Note that a new $N(t)$ value is obtained from the LS block on each time interval Δt. As shown in (12), for a fixed N value, the numerical value of p can still vary depending on I_{rst}. The smaller I_{rst}, the smaller the p is, therefore the larger $t_{ISI,max}$ is. This also explains why $t_{ISI,max}$ is defined by both I_{rst} and N from the theoretical perspective.

2) IV Converter

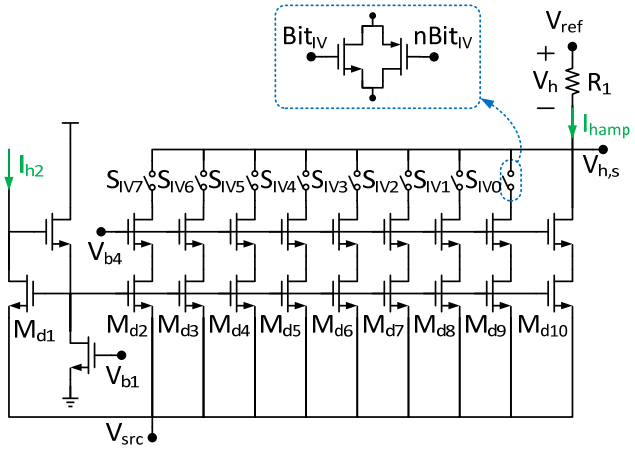


Fig. 8. CMOS circuit details of the IV Converter block

As mentioned in Section II.B, in order to generate random spikes, the value of h has to be compared to samples in the range of [0,1/Δt] drawn from a uniform distribution. These samples are represented as a voltage V_{nx}, which is updated on every Δt. The range of V_{nx} is defined from 0 to 1.25 V. On the chip, I_{h1} needs to be converted to a voltage through a given resistance R_{eq} for comparison to V_{nx}. However, implementing a physical linear resistor on the chip for I_{h1} is infeasible because of the large value of R_{eq}. For example, given a constant input N, the maximum hazard current of I_{h1} in (10) can be computed as

$$I_{h1,\max} = I_{h1}\left(t_{ISI\max}\right) = \left(\frac{I_{rst}}{128}\right)\cdot N \cdot ISI_{\max} \qquad (13)$$

This maximum value happens when V_h (= $I_{h1}\times R_{eq}$) reaches $V_{nx,max}$ whereupon a spike is generated. The ISI value carried by this spike is then t_{ISImax} (i.e. $ISI_{max}\times\Delta t$). If we set N = 64, I_{rst} = 256 pA, and ISI_{max} = 16, then according to (13), $I_{h1,max}$ = 2048 pA and the required R_{eq} as calculated by $V_{nx,max}/I_{h1,max}$, is equal to 610 MΩ.

However, the required resistance R_{eq} can be reduced by amplifying I_{h1}. The current splitter, mentioned in Section III.C.1) creates a current with an amplification of I_{h1}, by first making I_{h2} = 63×I_{h1}. Then I_{h2} is amplified further by 32x through a current mirror circuit leading to a resistor R_1 of only 340 kΩ, and that occupies a layout area of 71×28 um^2. To calibrate the resistance variation due to process mismatch, the output transistor of the 32x current mirror circuit is divided as shown in Fig. 8, into several transistor branches which can be turned on or off by switches S_{IV7} to S_{IV0}. The ratio of M_{d1} to M_{d10} is 4:128:64:32:16:8:4:2:1:1. $Bit_{IV}[7:0]$, shown in Fig. 6, represents the 8 bits to control the switches. The amplified current, I_{hamp}, is converted to a hazard voltage, V_h, by the resistor R_1. That is, $V_h = I_{hamp}\times R_1 = I_{h1}\times(63/4)\times Bit_{IV}\times R_1$, where the value of 63 is contributed by the current splitter in the **Hazard Core** block and the factor of 4 is from the transistor sizing ratio of M_{d1} to M_{d10}. Theoretically, R_{eq} = 63×32×R_1, where Bit_{IV} = 128, without considering process mismatch.

The output voltage, $V_{h,s}$ (= V_{ref}-V_h), is compared with $V_{nx,s}$, a voltage-shifted version of V_{nx}, in order to ease the design complexity of the RNG circuit and the comparator. $V_{nx,s}$ ranges from 1.25 V to 2.5 V.

3) Comparator

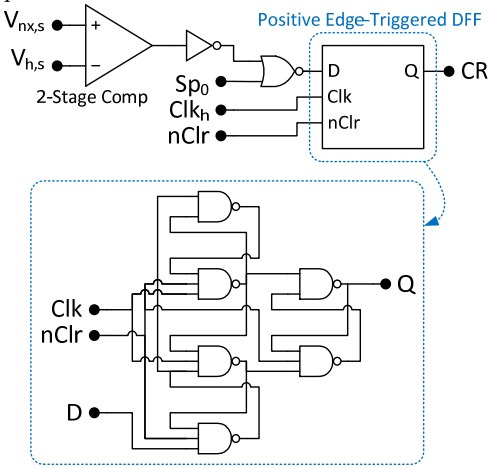


Fig. 9. Circuit details of the Comp block

The **Comp** block shown in Fig. 9 achieves three goals. First, it compares the hazard value $V_{h,s}$ to a random variable $V_{nx,s}$. Second, because the random samples are produced at every time step, Δt, the comparison result (CR) should be aligned with this time interval. Third, during the generation of the spike, the block should be disabled. That is, CR should stay high when there is a spike.

The first goal is achieved by a simple two-stage op-amp. Because $V_{h,s}$ (= V_{ref}-V_h) is inverted and has an offset from V_h, the comparison scheme also has to be modified. Theoretically, the compared result is high when $V_h > V_{nx}$. In reality, it happens when $V_{h,s} < V_{nx,s}$. Therefore, $V_{h,s}$ is connected to the negative terminal of the comparator as shown in Fig. 9. The second and third requirements are reached by implementing a positive edge-triggered D flip-flop with a clock, Clk_h, and a feedback spike pulse, Sp_0, as inputs. Clk_h is generated from the LS block with a period time the same as the time step, Δt. The disable period is determined by the pulse width of Sp_0, which is generated from the **Spike Generator & Channel AER** block. The default pulse width is set to 2×Δt.

4) Spike Generator & Channel AER

In this block as shown in Fig. 10, the channel AER communicates with the **Chip AER Transmitter** block through the $nReq$ and Ack signals. The $nReq$ signal becomes active if the output of the **Comp** block (CR) is high. The output of the de-multiplexer, Sp_0, has a pulse width that is determined by selecting one of the outputs of six JK flip-flops. These gates produce outputs that are multiples of either 1, 2, 4, 8, 16 or 32 periods of Clk_h. From our simulations, a pulse width of 2 clock periods is sufficient to allow the voltages and currents of the circuits in the **Hazard Core** block to return to their initial values.

TABLE I shows the mapping between the mathematical values and the hardware implementation in this work. Note that

the mapping of the probability distribution, p, as the hazard input is dependent on both N and I_{rst} as shown in (12). The relationship between the hazard value, h, and the hazard current, I_{h1}, is defined in (11).

Fig. 10. CMOS circuit details of the Spike Generator & Channel AER block

TABLE I: Mapping Table

Math. Symbol	Math. Value	Physical Symbol	Physical Value
p	[0,1]	$N\times I_{rst}$	$[1,64]\times I_{rst}$
-	-	I_{rst}	~300 (pA)
h	[0,1/Δt] = [0,1]	V_h	[0,1.25] (V)
-	-	$V_{h,s}$	[1.25,2.5] (V)
nx/Δt	[0,1/Δt] = [0,1]	V_{nx}	[0,1.25] (V)
-	-	$V_{nx,s}$	[1.25,2.5] (V)
Δt	1	Δt	64 (us)
-	-	pulse width of Sp_0	$2\times\Delta t$ = 128 (us)
-	-	Clk_h	$1/\Delta t$ = 15.6 (kHz)
-	-	Clk_{main}	10 (MHz)
-	-	Clk_{RNG}	0.5 (MHz)
-	-	Clk_{config}	1.56 (MHz)
-	-	C_1	4 (pF)
-	-	R_1	340 (kΩ)
-	-	V_{ref}	2.5 (V)
-	-	V_{src}	0.3 (V)

IV. RESULTS

The chip microphotograph is shown in Fig. 11. Because t_{ISImax} of each channel is controlled by I_{rst} (Section III.C.1), we implement 16 on-chip DACs that allow us to tune the I_{rst} value of each individual channel. The subsections below present the detailed chip characterization results and measurements from the entire system with both RS and LS blocks.

A. Accumulation of Hazard

We first measured the RS block. The chip data show how the hazard accumulates given a uniform distribution, which means a constant input N, as an example.

The hazard value h can be derived by measuring V_h. This voltage can be indirectly estimated by measuring the output ISIs of one RS channel as a function of V_{nx} as shown in Fig. 12.Note that in this case V_{nx} is set by an external voltage source. Because an output spike is generated when $V_h > V_{nx}$, the accumulation time of the hazard increases as V_{nx} increases.

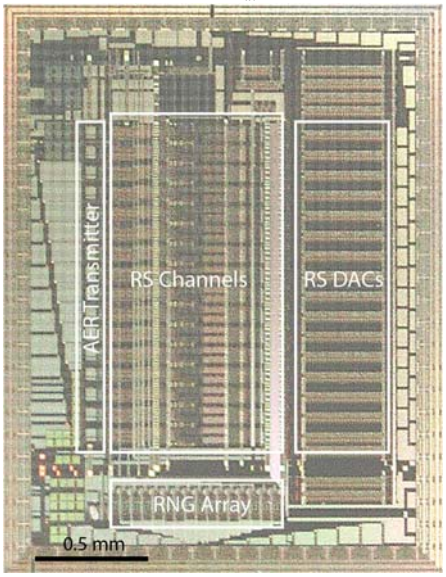


Fig. 11. Chip microphotograph of the RS array with 16 RS channels, 16 RS DACs for the reset currents, the RNG array for 16 random sources and the AER transmitter. The bias generator occupies the remaining area. The chip areas is 2.16×2.74 mm^2.

The value of V_h is equal to V_{nx} when a spike is generated. Therefore, Fig. 12 also demonstrates how the hazard accumulates over time in the case of a uniform input distribution. This can be seen by reversing the x and y axes of the plot. The ISI_{max} for each value of N happens when V_h reaches $V_{nx,max}$ (= 1.25 V). Note that in Fig. 12, the ISI_{max} value for N = 64 is 0.5 of the ISI_{max} value for N = 32 and so on. The experimental result follows the trend in (13) that $N\times ISI_{max}$ is a constant.

The inset of Fig. 12 shows an expanded view of the curves in the dotted box. Here, it is clear that using the smallest value of N corresponding to a larger t_{ISImax} (= $ISI_{max}\times\Delta t$) also means that the initial V_h is small and that the voltage changes very little over a large range of ISI values. To detect a small change in V_h means that a random source with high bit resolution is required. The required resolution will be discussed in the next section.

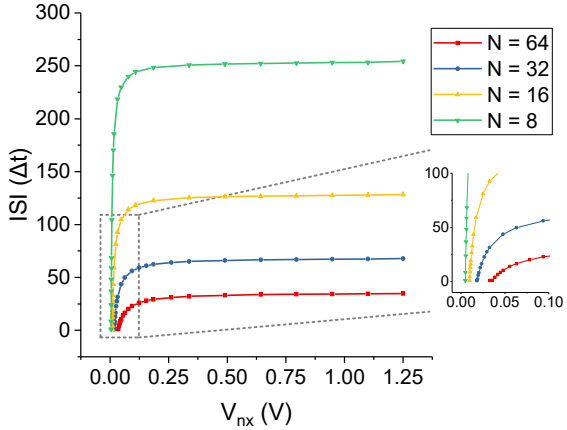


Fig. 12. Dependence of output ISIs on V_{nx}.

B. Effect of the Non-Ideal Random Source

In next experiment, we tested if the output ISI distribution of

the RS channel is affected by the non-ideal random source. We provide a uniform distribution by using a constant input N and measure the output ISI distribution by collecting the output ISIs.

We now consider $V_{nx}(t)$ as independent samples of a random source that are drawn on every Δt. Given a uniform input distribution, the output ISIs between 1 to ISI_{max} should be equiprobable. In the inset of Fig. 12, it can be seen that V_h increases slowly before the time reaches t_{ISImax}. During integration, V_h is continuously compared to V_{nx} so the random source needs a high resolution in order that small changes in V_h could be distinguished.

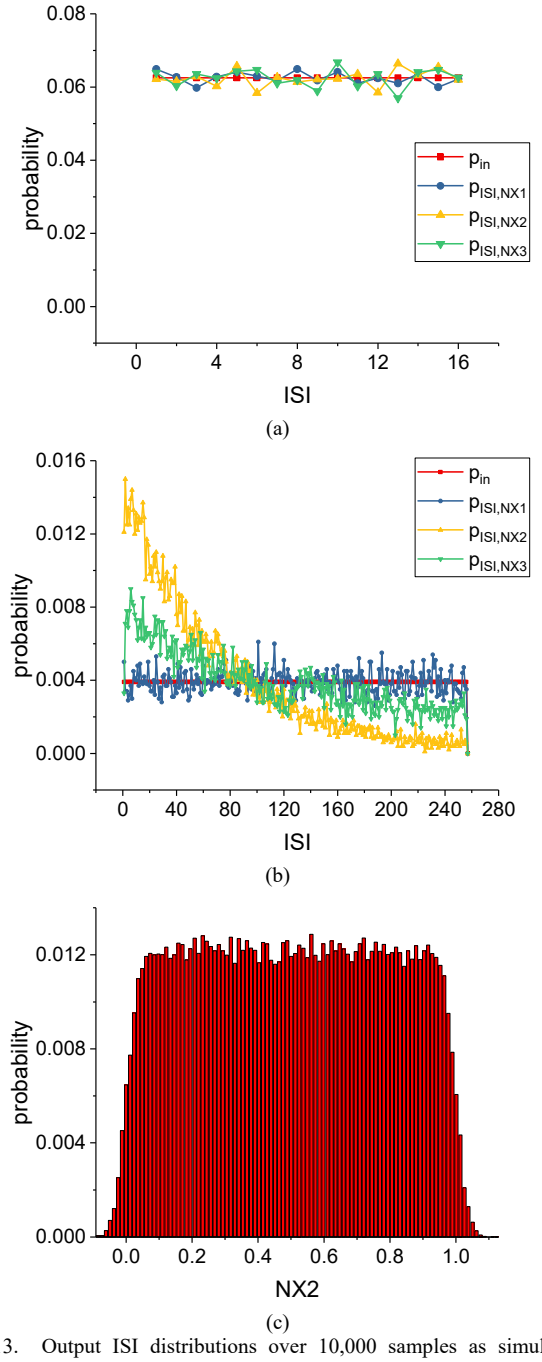


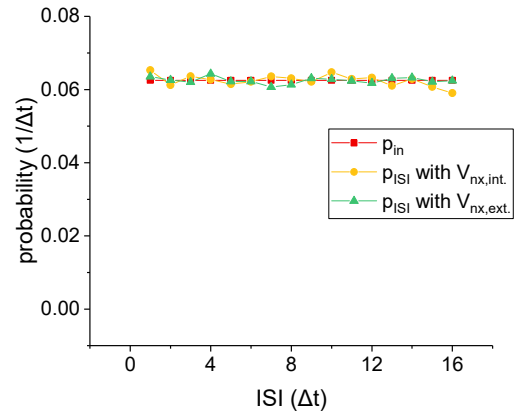
Fig. 13. Output ISI distributions over 10,000 samples as simulated in MATLAB. The uniform input probability p_{in} is (a) 0.0625 (b) 0.0039. (c) The distribution of the random source *NX2*.

Both the RNG and LFSR random sources applied on the chip are non-ideal: The RNG circuit has a non-ideal uniform distribution and the LFSR has a finite resolution in the distribution values. We verify if the non-idealities affect the resulting output ISI distribution. The distribution of RNG measured in [20] shows Gaussian distortion similar to Fig. 13(c). The LFSR as implemented in the FPGA can only have finite resolution. The effect of these non-idealities on the output ISI distribution are explored in a Matlab simulation for the three cases of the random source: An ideal uniform random source, *NX1*; a uniform random source, *NX2* (Fig. 13(c)), where a 0.03-sigma Gaussian random value is added to the samples, and a random source which has only a 8-bit resolution, *NX3*. The impact of these different random sources on the output distribution is dependent on the resolution of the ISI bins. In the case when $ISI_{max} = 16$, corresponding to $p_{in} = 0.0625$, the output distributions of all three sources as shown in Fig. 13(a) are very similar. However in the case of a large ISI_{max}, corresponding to an even smaller p_{in}, only the ISI distribution of *NX1* is similar to the input as shown in Fig. 13(b).

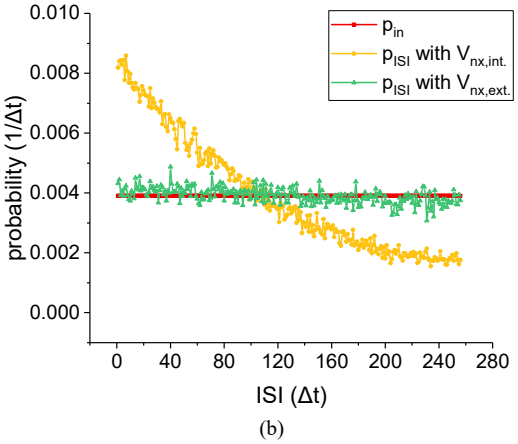
Fig. 14 shows measured test results from one RS channel of the fabricated chip. Note that I_{rst} is tuned so that $ISI_{max} = 16$ given a constant input $N = 64$ ($p_{in} = 0.0625$). The bit resolution of the external LFSR random source is set to 14 bits. The analog output after passing the LFSR output through a 14-bit off-chip DAC is used as the random number output instead of the internal RNG output. When ISI_{max} is small (= 16), the output distributions from using either the external or internal random sources looks similar to the input distribution as shown in Fig. 14(a). When ISI_{max} is increased to 256, the 14-bit LFSR in Fig. 14(b) performs much better than the on-chip RNG output that is similar to the simulation results in Matlab. The KL divergence $D(p||p_{out,n})$ characterizes the disparity between $p_{out,n}(i)$ and $p(i)$. This disparity is caused by the finite set of samples that are used by $p_{out,n}(i)$ for a representation of $p(i)$. The $p_{out,n}$ in our case is represented as the output ISI distribution, p_{ISI}, and the p is represented as the input distribution, p_{in}.

The KL divergence of the ISI output distributions generated by using the external (LFSR) and the internal (RNG) random sources are shown in Fig. 14(c). As expected, the value of the KL divergence increases following the increase of ISI_{max} given a fixed number of samples. However, the values using the internal random source is consistently higher across all ISI_{max} values. This trend suggests that the Gaussian distortion seen in the RNG distribution has a worse effect on the output ISI distribution than the finite bit resolution of the LFSR.

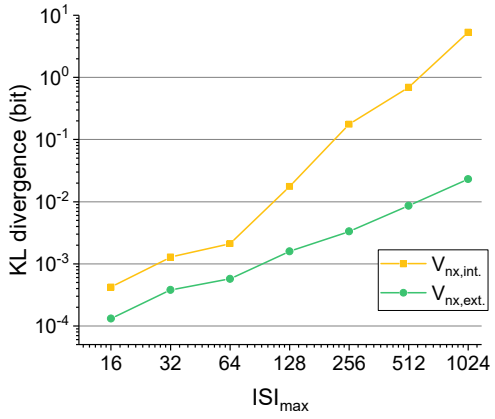
In this experiment, we see that the Gaussian distortion from the RNG affects the output distribution as ISI_{max} increases. We are unable to reduce this non-ideality in a simple way after fabrication. By contrast, the resolution of LFSR can be adjusted to an arbitrary number of bits if the corresponding DAC can be found. With a 14-bit LFSR and $ISI_{max} = 128$, we get a KL divergence of 0.002 as shown in Fig. 14(c). In the experiments described in the following subsections, the random sources are all provided from the LFSR external source.



(a)



(b)



(c)

Fig. 14. Output ISI distributions over 80,000 samples as measured from one RS channel using a constant input N (a) 64 and (b) 4. The corresponding mathematical input probability p_{in} is (a) 0.0625 (b) 0.0039. (c) The KL divergence of the output ISI distributions over the 80,000 samples with the internal (RNG) and external (LFSR) random sources.

C. Calibration of the Equivalent Resistance

In Section III.C.2) we showed how to compute the resistance R_{eq} from a special case of (13). In fact, R_{eq} can be theoretically formulated as in (14).

$$R_{eq} = \frac{\kappa_{M_{a1}}}{C_1 U_T} \cdot V_{nx,\max} \cdot \Delta t \qquad (14)$$

In this equation, the value of R_{eq} is set by $V_{nx,max}$ and Δt. We calibrate the ratio of the current mirror in the **IV Converter** block by $Bit_{IV}[7:0]$ so that the equivalent resistance of the circuit (= $R_1 \times (63/4) \times Bit_{IV}$) matches the theoretical R_{eq} in (14). Otherwise, the output ISI distribution does not approximate the input. Ideally, $Bit_{IV} = 128$. We demonstrate how the calibration affects the ISI output distribution, p_{ISI}, in Fig. 15. Given a uniform input distribution ($N = 32$), only p_{ISI} with $Bit_{IV} = 107$ is close to p_{in}. For those $Bit_{IV} < 107$, the RS channel tends to generate spikes at large ISIs because the equivalent resistance is not big enough. On the other hand, when $Bit_{IV} > 107$, more spikes are prone to be generated at small ISIs.

Because of circuit mismatch, the calibration has to be done separately in each channel. A memory block that stores the value for each channel would have been useful. Since we do not have this on-chip, we choose maximally 6 out of the 16 channels that have similar values of Bit_{IV}, to construct a factor graph.

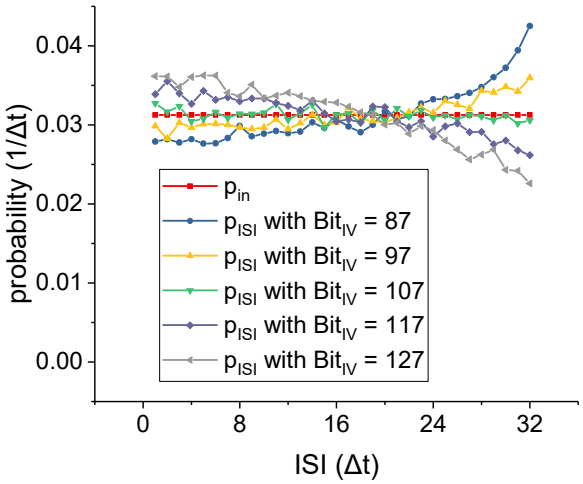


Fig. 15. Effect of the calibration on the output ISI distributions. Given a constant input N = 32, p_{ISI} is obtained from one RS channel over 80,000 samples. The corresponding mathematical input probability p_{in} is 0.031.

D. Message Passing in VLSI Factor Graphs

In these experiments, the results are obtained from the complete system. Each LS-RS-combined channel represents a unidirectional massage passing and several channels make up a unidirectional factor graph. The results on two factor graphs are presented here. Note that the total time window, W_T, is based on both the number of collected samples and the mean of the ISI distribution, ISI_{mean}. For example, if ISI_{mean} of a particular message is 16.5 and we want to collect 100,000 samples of this message, then the average W_T is $16.5 \times 100000 \times \Delta t = 105.6$ s for Δt = 64 us. The samples in the experiments are collected over sequential time windows, each of length W (= 1.05 s), meaning that the LS channel is reused several times. The **Mem ISIx** and **Mem ISIY** modules have 512 entries each. The output samples generated by the RS channel are based on the input-message-combined distribution, which is computed from the 512^2 sample pairs. The samples are stored in the **Mem Mz** module. At the start of each time window, all three **Mem** modules are reset.

The first factor graph shown in Fig. 16(a) is composed of three channels. Two of them carry ISI samples of X and Y drawn from uniform distributions m_X, and m_Y. The third channel carries the output of a switching gain function defined in (15).

I_{rst} is set so that the ISI_{max} values of m_X and m_Y are 32 and 16, respectively.

$$f(x,y,z)=\begin{cases}\delta(x-z) & \text{if } 1\le y\le 18\\ \delta(0.5x-z) & \text{if } 9\le y\le 16\end{cases} \quad (15)$$

The switching-gain node sets a gain of 1 on X if the ISIs of Y are between 1 and 8, and a gain of ½ if the ISIs of Y are between 9 and 16. Fig. 16(b)-(d) show the messages of variables X, Y, Z along the arrows. Since both X and Y ISI samples have uniform distributions, the value of m_Z for ISI <= 16 is approximately 3x the value when ISI > 16. In the first time window, the samples of m_X and m_Y just arrive at the switching-gain node so the samples of m_Z in this window are not used. Fig. 16(e) shows that the KL divergence of the three messages as computed from the theoretical and measured output ISI distributions as a function of the number of ISI samples. Because the W_T values are different for the three messages due to different ISI_{mean} (see caption of Fig. 16), the number of time windows needed in order to obtain the same number of samples for all three messages are different.

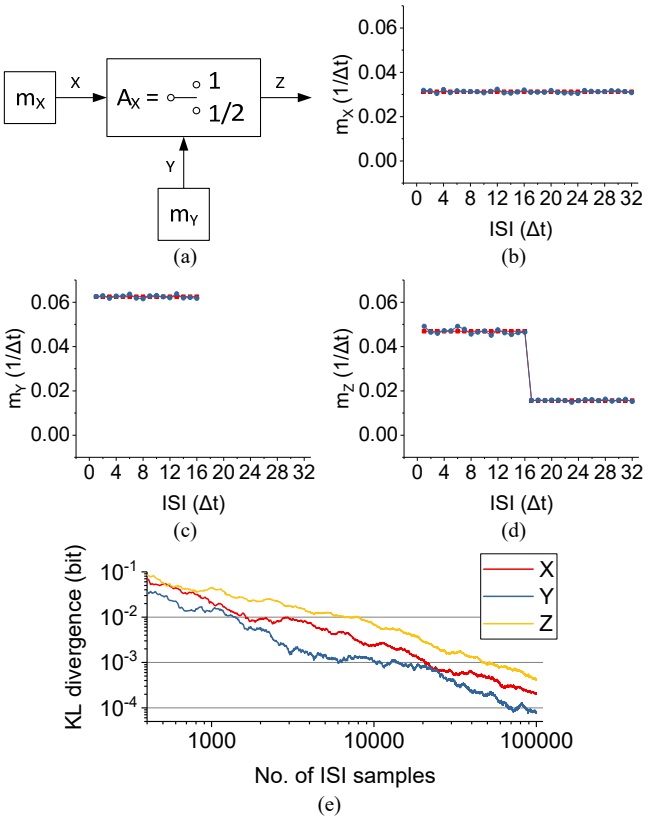


Fig. 16. (a) Factor graph consisting of three channels, three variables X, Y, Z and the messages along the arrows. The messages of (b) X (c) Y (d) Z with the theoretical distribution in red and the output ISI distribution (in blue) over 100,000 samples. Because of the different ISI_{mean} values of 16.5, 8.5 and 12.5 for these messages, the corresponding W_T values are 105.6 s, 54.4 s and 80 s. (e) KL divergence of the messages as a function of the number of ISI samples.

The second factor graph shown in Fig. 17(a) consists of six channels. m_U and m_V have the same uniform distribution with ISI_{max} = 16 and m_W is defined like a v-shape as shown in Fig. 17(b). The functions of the '+' and '=' nodes are defined in (6) . The node after '=' is defined as a half-wave Rectified Linear Unit (ReLU) with an ISI threshold of 10 as shown in (16).

$$f_{rectifier}(y,z)=\begin{cases}0 & \text{if } y\le 10\\ \delta(y-z-10) & \text{if } y>10\end{cases} \quad (16)$$

The plots in Fig. 17(b)-(e) show the messages of W, X, Y, Z along the arrows. Because of the uniform distributions of m_U and m_V, the distribution of m_X has the expected shape with ISI_{max} = 32. Then, after the '=' node, the distribution of m_Y is the product of m_W and m_X. Finally, the rectifier produces m_Z which is a shifted version of m_Y. Fig. 17(f) shows the respective KL divergence of the messages as computed from the theoretical and measured output ISI distributions as a function of the number of ISI samples. Similar to the switching-gain example, only the samples of m_Z after the third time window are used because the valid samples of m_X arrive at the equality constraint node only after the second time window.

The KL divergence values of messages m_Y and m_Z are generally higher than that of messages m_W and m_X. This is because the equality constraint node filters out many input sample pairs, (ISI_x,ISI_w) following (6). If the ISIs of any possible pairings of (ISI_x,ISI_w) are not equal, these pairs are discarded. In our experience, many such pairs exist. The histogram formed from the remaining sample pairs is used to generate the resulting output spikes whose ISI distribution has a worse approximation to the theoretical distribution than other constraint functions, such as function f_{plus} where samples are not discarded. This approximation error propagates to the next factor node for Z.

Because in the initial state of the system, the Mem ISI, Mem Hist_Z and Mem M_Z block values are set to zero, there will be a transient response, during which the distribution would be incorrect. In a case with stationary inputs, we could wait for a short time before using the probability distributions. In the plots of Fig. 17, we show the response after the system has settled, in this case, for inputs with fixed distributions. For a time-varying input, the change in the distribution should be slower than W so that the output spike train reflects the correct distribution. If there are multiple time-varying inputs, additional constraint nodes will be needed to delay spike trains arriving to certain nodes so that the output distribution is computed properly. For example, in Fig. 17, if m_u, m_v, and m_w are changing, we can add a unity-gain constraint node (see (6) and A = 1) between m_w and the equality constraint node to delay m_w. Therefore, the spike trains reflecting the correct distributions will arrive at the equality constraint node within the same time window.

The system specifications are given in TABLE II. The power consumption of the entire chip includes the power of the 16 on-chip DACs which are needed to adjust the reset current, I_{rst}, of each channel separately.

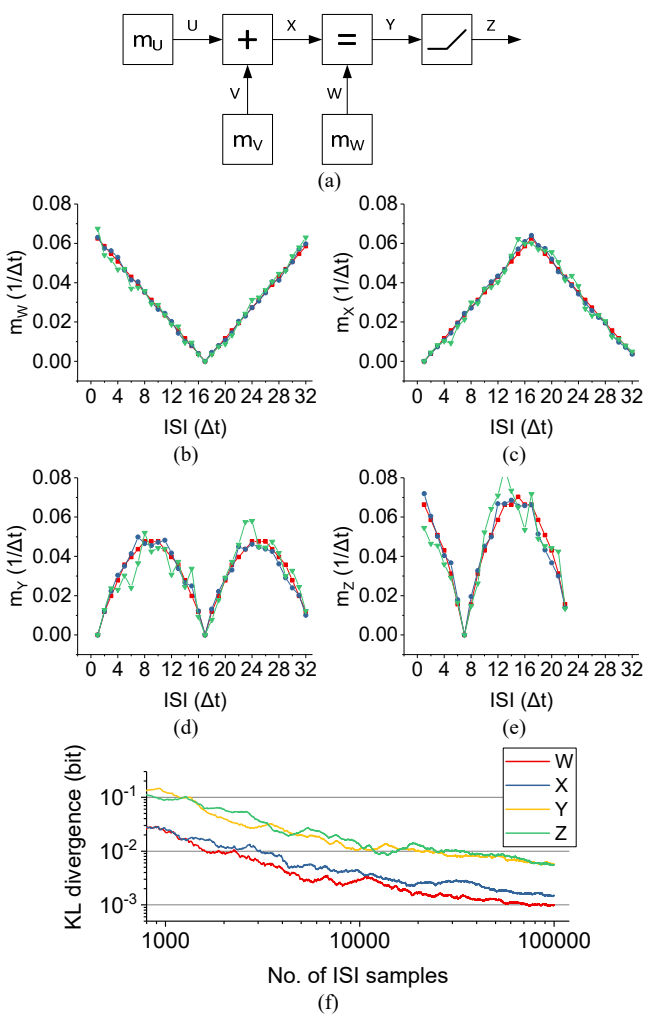


Fig. 17. (a) Factor graph consisting of six channels and variables U, V, W, X, Y, Z, and the messages passing along the arrows. The messages of (b) W (c) X (d) Y (e) Z with the theoretical distribution in red and the output ISI distribution over 4,000 (in green) and 100,000 samples (in blue). Because of the different ISI_{mean} values of 16, 17, 17 and 11.8, the corresponding W_T values for 100,000 samples are 102.4 s, 108.8 s, 108.8 s and 73.5 s. (f) KL divergence of the messages as a function of the number of ISIs.

TABLE II: System Specifications

Specification	Quantity
Process	AMS 2P4M 0.35 um
Chip Area (mm^2)	2.16 × 2.74
Chip Area (One RS Channel) (mm^2)	0.78 × 0.09
Number of Channels	16
Power of the RS chip (mW)	6.32
Power of the RS single channel (mW)	0.046
Supply Voltage (V)	3.3
System Clock Clk_{main} (MHz)	10
System Clock Clk_h (kHz)	15.6
Δt (us)	64
Time Window W (s)	1.05
ISI Range (Δt)	1 to 128
RS Input Range	0 to 63
Reset Current I_{rst} (pA)	~300
Random Source $V_{nx,s}$ Range (V)	1.25 to 2.5
LFSR Resolution (bits)	14

V. DISCUSSION

In this work, we presented a hardware system that implements a set of output message channels in a factor graph model, where the belief-propagation messages are transmitted in an event-based fashion. The system consists of two components: The first component is an ASIC chip that implements a unidirectional set of factor graph nodes that can produce up to 16 output messages in the form of spike trains. The second component is an FPGA system that records the ISI distribution from the spike trains and the functions of the factor nodes. The messages for the opposite direction between two nodes can be computed by reusing the output channels on the ASIC chip. Results from this hardware system show that the distributions of the inputs and output variables are good approximations of the predicted theoretical distributions as measured by the KL divergence.

The number of operations and memory needed for our event-based hardware model and a hypothetical SPR hardware are discussed next. Assuming that ISI_{max} = 128 and the constraint function is limited to a delta function as in the examples of Section IV.D, and $W_T = 100 \times W$, the event-based model needs 26.2E6 (= 100×512^2) additions and a simple bit shift for the normalization. No multiplication is needed. On the other hand, the SPR computation will need 2.1E6 (= 128^3) additions, 16.4E3 (128^2) multiplications, and 1 division for the normalization term. If fewer samples can be used (e.g. 4,000 samples) for the approximation in the event-based model, then the number of additions can be reduced to 1.05E6. In terms of input/output memory size, our event-based model requires 7936 (= 2×7×512+6×128) bits while a factor node using the SPR requires 768 (= 6×128) bits. Even though the event-based model needs a larger memory, it does not need the multipliers and the divider required for the SPR. If the SPR hardware computes all messages in parallel, each factor node still requires the multiplier and divider circuits. For the simple factor graph examples shown in the paper, the operators of the SPR are not a big cost. However, the complexity of the hardware for the event-based model is lower if one were to scale up the graphs. In addition, the current FPGA plus ASIC system can be integrated into a future ASIC.

This hardware system differs from previous analog VLSI factor graphs that implement belief-propagation algorithms [30], [31]. Besides the clear difference in which the messages are encoded, the circuits in [30] work with factor entries of 0 and 1 only and uses binary variables. The work of [31] shows the implementation of a higher-order factor node but the authors describe only a single instance of a factor node.

There are other ways of performing approximate inference in graphical models through spiking neurons. One proposed method is that of neural sampling [32]–[35] in a graph with binary nodes and where the nodes are represented by spiking neurons. This form of sampling belongs to the MCMC technique and is similar to Gibbs sampling. This sampling scheme has been demonstrated on the SpiNNaker hardware system which has an architecture of fixed-point ARM9 cores [36], [37].

Similar to other stochastic models such as RBMs, our event-based belief propagation VLSI model requires a pseudo-number generator for the sampling process. Several aVLSI

implementations of pseudo-number generators have been proposed over the years and have focused on either digital noise (noisy bits) [38], [39], uniform distributions [40], or other types of predefined, fixed distributions [41]–[43], or true random number generator circuits which generate discrete, Bernoulli and quasi-continuous, exponential random variables [44]. In [45], a single-photon avalanche diode (SPAD) was used as aVLSI noise source and connected to a spike-response neuron model [25] implemented on an FPGA. Digital random number generators have been used as a cheap way of creating random connections for a network using the Neural Engineering Framework [46]. In our work, we used an on-chip pseudo-random generator [27] previously implemented in [20], but because of the non-ideal Gaussian distortion of the output distribution caused by the switched capacitors, we used an LFSR implemented on the FPGA for the results reported in this work. A better random number generator circuit can be designed through larger component sizes. A further option is to use a current-mode random number generator circuit so that the I-V converter is not required. However, similarly as in the circuit we used, one would need to amplify the hazard current I_{h1} so that the time constant of the inputs to the subsequent comparator circuit is reduced.

Because of the clocked nature of either the on-chip SPR circuit or the LFSR circuit, the ISI samples in our implementation are limited to integer values. However this limitation allows us to design a simple counter for this range of integer values and therefore to compute the histograms of two functions $f(x,y,z)$ and $F(x,y)$ in Fig. 5 in parallel. The ISI integer values in this work were chosen to go from 1 to 128 (t_{ISI} goes from 64 us to 128×64 us). To increase ISI_{max}, an on-chip random number generator with a more ideal output distribution is needed and the subthreshold transistor representing the exponential term in (10) will need to cope with the corresponding large range of currents.

The value of ISI_{max} determines the ISI samples needed to approximate a distribution. However, the required number of samples also depends on the task. For example, in an object tracking task, the probability distribution of the target's position is often similar to a Gaussian distribution which can be approximated using fewer samples in contrast to the case of the uniform distribution. The system speed can be increased by increasing I_{rst} and decreasing Δt with the tradeoff of increased power consumption. For example, if the initial current is set to $10I_{rst}$ and the time step is set to $0.1\Delta t$, the time required for collecting 100,000 samples decreases to 10.56 s while the power consumption in a RS channel increases to 0.46 mW instead of 0.046 mW. In addition, the lengthy accumulation time can be reduced by combining multiple input streams together into one input stream to a factor node. This accumulation time can also be reduced by having multiple factor nodes representing one distribution. The time required to collect 100,000 samples, for example, reduces to 10.56 s with 10 factor nodes. The tradeoff is that 10 RS channels will be required leading to a power dissipation of 0.46 mW.

A future extension of this work would be to configure this system to implement a network that performs inference on the output of event-based sensors such as the DVS [47] and the cochlea which generates asynchronous outputs [48]. For example, this hardware system can be used together with the DVS in a tracking task such as predicting the position of an object across the field of view of the retina. We could also implement a Kalman filter for this object tracking task. Instead of using the mean and variance, our event-based model encodes the information about the prediction and observation states within the distributions carried by the spike trains. In this case, the DVS events would serve as the observation input. The plus, gain, and equality constraint nodes capture the remaining operations of the Kalman filter. This event-based model therefore encodes the probability of the position of the object in the image and in addition, the distribution need not be only Gaussian.

ACKNOWLEDGEMENT

The authors thank members of the Sensors group at the Institute of Neuroinformatics and Minhao Yang for circuit discussions.

REFERENCES

[1] T. Pfeil *et al.*, "Six networks on a universal neuromorphic computing substrate," *Front. Neurosci.*, vol. 7, 2013.

[2] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker project," *Proc. IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

[3] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.

[4] N. Qiao *et al.*, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses," *Front. Neurosci.*, vol. 9, 2015.

[5] B. V. Benjamin *et al.*, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proc. IEEE*, vol. 102, no. 5, pp. 699–716, May 2014.

[6] D. Neil and S. C. Liu, "Minitaur, an event-driven FPGA-based spiking network accelerator," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 22, no. 12, pp. 2621–2628, Dec. 2014.

[7] T. J. Hamilton, S. Afshar, A. van Schaik, and J. Tapson, "Stochastic electronics: A neuro-inspired design paradigm for integrated circuits," *Proc. IEEE*, vol. 102, no. 5, pp. 843–859, May 2014.

[8] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Front. Neurosci.*, vol. 7, 2013.

[9] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 x 128 120 dB 15 us latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.

[10] S. C. Liu, A. van Schaik, B. A. Minch, and T. Delbruck, "Asynchronous binaural spatial audition sensor with

2x64x4 channel output," *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 4, pp. 453–464, Aug. 2014.

[11] E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs, "Event-driven contrastive divergence for spiking neuromorphic systems," *Front. Neurosci.*, vol. 7, 2014.

[12] B. U. Pedroni *et al.*, "Mapping generative models onto a network of digital spiking neurons," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 4, pp. 837–854, Aug. 2016.

[13] K. Sanni, G. Garreau, J. L. Molin, and A. G. Andreou, "FPGA implementation of a deep belief network architecture for character recognition using stochastic computation," in *2015 49th Annual Conference on Information Sciences and Systems (CISS)*, 2015, pp. 1–5.

[14] A. Steimer and R. Douglas, "Spike-based probabilistic inference in analog graphical models using interspike-interval coding," *Neural Comput.*, vol. 25, no. 9, pp. 2303–2354, Sep. 2013.

[15] H. A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F. R. Kschischang, "The factor graph approach to model-based signal processing," *Proc. IEEE*, vol. 95, no. 6, pp. 1295–1322, Jun. 2007.

[16] H. A. Loeliger, "An introduction to factor graphs," *IEEE Signal Process. Mag.*, vol. 21, no. 1, pp. 28–41, Jan. 2004.

[17] J. Y. Shih, C. A. Atencio, and C. E. Schreiner, "Improved stimulus representation by short interspike intervals in primary auditory cortex," *J. Neurophysiol.*, vol. 105, no. 4, pp. 1908–1917, Apr. 2011.

[18] W. M. Usrey, J. B. Reppas, and R. C. Reid, "Paired-spike interactions and synaptic efficacy of retinal inputs to the thalamus," *Nature*, vol. 395, no. 6700, pp. 384–387, Sep. 1998.

[19] W. M. Usrey, J.-M. Alonso, and R. C. Reid, "Synaptic interactions between thalamic inputs to simple cells in cat visual cortex," *J. Neurosci.*, vol. 20, no. 14, pp. 5461–5467, Jul. 2000.

[20] C. H. Chien, S. C. Liu, and A. Steimer, "A neuromorphic VLSI circuit for spike-based random sampling," *IEEE Trans. Emerg. Top. Comput.*, vol. PP, no. 99, pp. 1–1, 2016.

[21] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Stat. Comput.*, vol. 10, no. 3, pp. 197–208, Jul. 2000.

[22] O. Cappe, S. J. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential Monte Carlo," *Proc. IEEE*, vol. 95, no. 5, pp. 899–924, May 2007.

[23] F. Rieke, D. Warland, R. de R. van Steveninck, and W. Bialek, *Spikes: Exploring the Neural Code*, Reprint edition. A Bradford Book, 1999.

[24] M. E. Larkum, J. J. Zhu, and B. Sakmann, "Dendritic mechanisms underlying the coupling of the dendritic with the axonal action potential initiation zone of adult rat layer 5 pyramidal neurons," *J. Physiol.*, vol. 533, no. Pt 2, pp. 447–466, Jun. 2001.

[25] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[26] K. A. Boahen, "A burst-mode word-serial address-event link-I: Transmitter design," *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 51, no. 7, pp. 1269–1280, Jul. 2004.

[27] G. Cauwenberghs, "Delta-sigma cellular automata for analog VLSI random vector generation," *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.*, vol. 46, no. 3, pp. 240–250, Mar. 1999.

[28] S.-C. Liu, J. Kramer, G. Indiveri, T. Delbruck, and R. Douglas, *Analog VLSI: Circuits and Principles*. Cambridge, Mass: A Bradford Book, 2002.

[29] B. Linares-Barranco and T. Serrano-Gotarredona, "On the design and characterization of femtoampere current-mode circuits," *IEEE J. Solid-State Circuits*, vol. 38, no. 8, pp. 1353–1363, Aug. 2003.

[30] H. A. Loeliger, F. Lustenberger, M. Helfenstein, and F. Tarkoy, "Probability propagation and decoding in analog VLSI," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 837–843, Feb. 2001.

[31] S. Hemati and A. H. Banihashemi, "Iterative decoding in analog CMOS," in *Proceedings of the 13th ACM Great Lakes Symposium on VLSI*, New York, NY, USA, 2003, pp. 15–20.

[32] D. Pecevski, L. Buesing, and W. Maass, "Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons," *PLOS Comput. Biol.*, vol. 7, no. 12, p. e1002294, Dec. 2011.

[33] L. Buesing, J. Bill, B. Nessler, and W. Maass, "Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons," *PLoS Comput. Biol.*, vol. 7, no. 11, p. e1002211, Nov. 2011.

[34] W. Maass, "Noise as a resource for computation and learning in networks of spiking neurons," *Proc. IEEE*, vol. 102, no. 5, pp. 860–880, May 2014.

[35] M. A. Petrovici, J. Bill, I. Bytschok, J. Schemmel, and K. Meier, "Stochastic inference with deterministic spiking neurons," *ArXiv13113211 Cond-Mat Physicsphysics Q-Bio Stat*, Nov. 2013.

[36] D. R. Mendat, S. Chin, S. Furber, and A. G. Andreou, "Neuromorphic sampling on the SpiNNaker and parallella chip multiprocessors," in *2016 IEEE 7th Latin American Symposium on Circuits Systems (LASCAS)*, 2016, pp. 399–402.

[37] D. R. Mendat, S. Chin, S. Furber, and A. G. Andreou, "Markov chain Monte Carlo inference on graphical models using event-based processing on the spinnaker neuromorphic architecture," in *2015 49th Annual Conference on Information Sciences and Systems (CISS)*, 2015, pp. 1–6.

[38] W. T. Holman, J. A. Connelly, and A. B. Dowlatabadi, "An integrated analog/digital random noise source," *IEEE Trans. Circuits Syst. Fundam. Theory Appl.*, vol. 44, no. 6, pp. 521–528, Jun. 1997.

[39] M. Bucci and R. Luzzi, "Fully digital random bit generators for cryptographic applications," *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 55, no. 3, pp. 861–875, Apr. 2008.

[40] C.-C. Wang, J.-M. Huang, H.-C. Cheng, and R. Hu, "Switched-current 3-bit CMOS 4.0-MHz wideband random signal generator," *IEEE J. Solid-State Circuits*, vol. 40, no. 6, pp. 1360–1365, Jun. 2005.
[41] P. Xu, Y. l Wong, T. K. Horiuchi, and P. A. Abshire, "Compact floating-gate true random number generator," *Electron. Lett.*, vol. 42, no. 23, pp. 1346–1347, Nov. 2006.
[42] P. Dudek and V. D. Juncu, "An area and power efficient discrete-time chaos generator circuit," in *Proceedings of the 2005 European Conference on Circuit Theory and Design, 2005.*, 2005, vol. 2, p. II/87-II/90 vol. 2.
[43] J. Holleman, S. Bridges, B. P. Otis, and C. Diorio, "A 3 uW CMOS true random number generator with adaptive floating-gate offset cancellation," *IEEE J. Solid-State Circuits*, vol. 43, no. 5, pp. 1324–1336, May 2008.
[44] B. Marr and J. Hasler, "Compiling probabilistic, bio-inspired circuits on a field programmable analog array," *Front. Neurosci.*, vol. 8, 2014.
[45] T. Clayton *et al.*, "An implementation of a spike-response model with escape noise using an avalanche diode," *IEEE Trans. Biomed. Circuits Syst.*, vol. 5, no. 3, pp. 231–243, Jun. 2011.
[46] R. Wang, C. S. Thakur, G. Cohen, T. J. Hamilton, J. Tapson, and A. van Schaik, "Neuromorphic hardware architecture using the neural engineering framework for pattern recognition," *IEEE Trans. Biomed. Circuits Syst.*, vol. 11, no. 3, pp. 574–584, Jun. 2017.
[47] R. Berner, C. Brandli, M. Yang, S. C. Liu, and T. Delbruck, "A 240x180 10mW 12us latency sparse-output vision sensor for mobile applications," in *2013 Symposium on VLSI Circuits*, 2013, pp. C186–C187.
[48] M. Yang, C. H. Chien, T. Delbruck, and S. C. Liu, "A 0.5V 55uW 64x2-channel binaural silicon cochlea for event-driven stereo-audio sensing," in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, 2016, pp. 388–389.

Chen-Han Chien (S'13) received the B.S. and M.S. in electrical engineering from National Tsing Hua University (NTHU), Taiwan in 2006 and 2008, respectively. He is currently working toward the Ph.D. degree at the Institute of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland. His research interests include probabilistic neural computation, neuromorphic VLSI design, low-power event-based visual and auditory sensors, and spike coding and processing.

Luca Longinotti received the B.Sc. degree in computer science from the University of Zürich, Switzerland. He currently works at iniLabs as a R&D Software Engineer, where he's responsible for FPGA logic, firmware, and low-level software development. His main interests lie in low-level and embedded software systems.

Andreas Steimer studied microsystems technology and physics at the Albert-Ludwigs University, Freiburg / Germany and received his Ph.D degree from the Institute of Neuroinformatics, ETH Zürich in 2012. He continued as a postdoc at the same institute until 2013. After that, he joined the Inselspital university hospital, Bern / Switzerland where he was a researcher until 2017. He is currently working as a machine learning engineer at the Bosch Center for Artificial Intelligence. His research interests include neuromorphic aVLSI circuit design, theoretical neuroscience and machine learning, particularly graphical models and message-passing algorithms.

Shih-Chii Liu (M'02–SM'07) studied electrical engineering as an undergraduate at the Massachusetts Institute of Technology and received the Ph.D. degree in the computation and neural systems program from the California Institute of Technology in 1997. She worked at various companies including Gould American Microsystems, LSI Logic, and Rockwell International Research Labs. She is currently a group leader at the Institute of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland. Her research interests include neuromorphic visual and auditory sensors, cortical processing circuits, and event-driven circuits, networks, and algorithms. Dr. Liu is past Chair of the IEEE CAS Sensory Systems and Neural Systems and Applications Technical Committees. She is current Chair of the IEEE Swiss CAS/ED Society and an associate editor of the IEEE Transactions of Biomedical Circuits and Systems and Neural Networks journal. She is a member of the IEEE Circuits and Systems Distinguished Lecturer Program for 2016-2017.